KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Testing probabilistic models of choice using column generation

Smeulders B, Davis-Stober C, Regenwetter M, Spieksma F.

KBI_1703

Testing Probabilistic Models of Choice using Column Generation*

Bart Smeulders† Clintin Davis-Stober‡
Michel Regenwetter§ Frits C.R. Spieksma¶

Abstract

In so-called random preference models of probabilistic choice, a decision maker chooses according to an unspecified probability distribution over preference states. The most prominent case arises when preference states are linear orders or weak orders of the choice alternatives. The literature has documented that actually evaluating whether decision makers' observed choices are consistent with such a probabilistic model of choice poses computational difficulties. This severely limits the possible scale of empirical work in behavioral economics and related disciplines. We propose a family of column generation based algorithms for performing such tests. We evaluate our algorithms on various sets of instances. We observe substantial improvements in computation time and conclude that we can efficiently test substantially larger data sets than previously possible.

1 Introduction

We consider computational challenges that arise when testing a certain type of probabilistic models of choice behavior. Imagine a decision maker who must specify a best element out of a set of distinct alternatives. In such situations, decision makers do not consistently select the same alternative as best, even when presented with the same (or nearly the same) set of alternatives (see for example Tversky [28]). Thus, assuming that a decision maker acts deterministically using a single decision rule (say, some linear order of the alternatives) is unrealistic. Probabilistic models of choice, pioneered by Block and Marschak [2] and Luce [18], attempt to explain uncertainty and fluctuations in behavior through probabilistic specifications. We concentrate on a class of models in which the permissible preference states are linear orders or weak orders of the alternatives. The random preference model captures the decision maker's uncertainty about preference with a probability distribution over these preference states. Then, for any given choice of the decision maker, the probability of choosing a particular alternative is governed by that probability distribution over preference states.

*This paper is based on a chapter in the PhD thesis of the first author; this research is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. This work is also supported by National Science Foundation grants SES-14-59866, PI: Davis-Stober & SES-14-59699, PI:Regenwetter. The authors also acknowledge support from the partnership between KU Leuven and the University of Illinois. We thank Prof. Yves Crama for discussions concerning this topic. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies or of the authors' universities.

†QuantOM, HEC Management School, Université de Liège, Rue Louvrex 14, B-4000 Liège, Belgium. The author is a post-doctoral fellow of the F.R.S-FNRS. Email: bart.smeulders@ulg.ac.be.

‡Department of Psychological Sciences, University of Missouri, 219 McAlester Hall, Columbia, MO 65211, USA. Email: stoberc@missouri.edu

§Department of Psychology, University of Illinois at Urbana-Champaign, 603 E. Daniel St., Champaign, IL 61820, USA. Email: regenwet@illinois.edu

¶ORSTAT, Faculty of Business and Economics, KU Leuven, Naamsestraat 69, B-3000 Leuven, Belgium. Email: frits.spieksma@kuleuven.be.

In a seminal contribution, McFadden and Richter [21] provide several equivalent (sets of) conditions for choice probabilities to be consistent with such a probabilistic model of choice. However, actually checking these conditions on choice probabilities poses computational challenges. Indeed, straightforwardly evaluating the "axiom of revealed stochastic preference" and the "Block-Marschak polynomials" both require checking a number of conditions that is exponential in the number of choice alternatives. Likewise, the system of linear inequalities and the linear programs given in McFadden and Richter [21] contain one variable for every preference state. The resulting number of variables grows exponentially in the number of alternatives, for most classes of preference states, including for linear orders. Even so, this linear programming model forms the basis of our column generation approach.

Most work on these probabilistic models has been on models *induced by linear orders* in *binary choice* settings. More precisely, the probability that a person chooses an alternative i over an alternative j, when required to choose one of the two, is the marginal probability of all linear orders in which i is preferred to j. Block and Marschak [2] described two classes of inequalities and proved that these inequalities are necessary and sufficient conditions for consistency with the probabilistic model of choice for data sets with up to 3 choice alternatives. Dridi [8] proved that these conditions are also necessary and sufficient for data sets with up to 5 alternatives and showed that they are no longer sufficient for data sets with 6 or more alternatives. Megiddo [22] proved that testing data sets for consistency with probabilistic choice induced by linear orders is difficult in general. He showed that the problem is equivalent to testing membership of a given point in the *linear ordering polytope*. Since optimization and separation over a particular polytope are polynomially equivalent (see Grötschel et al. [12]), it follows that testing whether a given collection of choice probabilities is consistent with a probabilistic model of choice induced by linear orders is NP-COMPLETE. In the last decades, researchers have generated extensive knowledge on the facial description of the linear ordering polytope (see Doignon et al. [7], Fiorini [10], the survey by Charon and Hudry [4], and the book by Martí and Reinelt [19], as well as the references contained therein).

When carrying out tests of probabilistic models of choice, scholars usually circumvent the computational challenges that arise when the number of alternatives grows large. Human laboratory experiments keep the number of alternatives small (see e.g., Cavagnaro and Davis-Stober [3] and Regenwetter et al. [25, 26], who use sets of 5 alternatives). Kitamura and Stoye [14] test a probabilistic version of the "strong axiom of revealed preference," using data from the U.K. Family Expenditure Survey, which they partition into subsets of a manageable size. While testing probabilistic choice models is difficult in general, it becomes easy for some settings and classes of preference states. Matzkin [20] and Hoderlein and Stoye [13] provide conditions for a probabilistic version of the so-called "weak axiom of revealed preference." Davis-Stober [6] describes a set of linear inequalities that are necessary and sufficient conditions for probabilistic choice induced by certain heuristic preferences. Smeulders [27] provides necessary and sufficient conditions for a probabilistic model induced by single-peaked linear orders. For all three of these settings, the conditions can be tested in polynomial time.

Here, we propose a family of algorithms based on column generation to test various probabilistic models of choice and apply it to a model induced by linear orders. Column generation is a technique to efficiently solve linear programs with a large number of variables; we come back to this technique in Section 3. Our main contribution is as follows:

Traditionally, the technique of column generation has almost always been applied to optimization problems. Here, however, we use column generation for a decision problem, namely, to detect whether given choice probabilities satisfy the probabilistic model of choice or not (i.e., a yes/no answer). We show how this affects the algorithm.

The rest of this paper unfolds as follows. In Section 2, we lay out the notation, the definitions and the model that we use. Section 3 provides a basic description of the column generation algorithms. Section 4 discusses the implementation of a family of such algorithms and reviews results from computational experiments. In Section 5 we show that when testing the model for many similar choice probabilities, the column generation algorithm can use output from one test to speed up subsequent tests. We illustrate how this is useful for statistical analysis of probabilistic

models, e.g., for calculating the Bayes factor to evaluate statistical performance on laboratory data from human subjects. Finally, we conclude in Section 6.

2 Notation and Definitions

Consider a set A, consisting of n many alternatives and let $A \star A = \{(i,j) \mid i \in A, j \in A, i \neq j\}$ denote the collection of all ordered pairs of distinct elements of A. For each ordered pair of distinct alternatives $(i,j) \in A \star A$, we are given a nonnegative number $p_{i,j} \leq 1$. These numbers represent the probabilities that i is chosen over j for all distinct i and j in A. For now, we concentrate on *two-alternative forced choice*, that is, the case in which a person must choose one alternative or the other when offered a pair of alternatives. Therefore, $p_{i,j} + p_{j,i} = 1$ for each pair of $i, j \in A$, $i \neq j$. We refer to such a collection $\{p_{i,j} \mid (i,j) \in A \star A\}$ of binary choice probabilities as a *data set*. We denote a preference order over the alternatives by the relation $\succ$ and we use the index m to indicate a particular preference order. If, for the preference order $\succ_m$, the alternative $i \in A$ is preferred over the alternative $j \in A$, we write $i \succ_m j$. The relations $\succ_m$ are asymmetric, complete and transitive. The set of all such preference orders is O. We further consider the subsets $O_{i,j} \subset O$ for each $(i,j) \in A \star A$, where each $O_{i,j}$ contains all preference orders $\succ_m$ in which $i \succ_m j$. The particular probabilistic model of choice that we use is called the *mixture model* (also known as random preference model): this model assumes that when a decision maker is faced with a choice, each preference order has a certain probability of governing the choice. When these probabilities are consistent with the numbers $p_{i,j}$, we say that the mixture model rationalizes the data set.

Definition 1. *Choice probabilities $\{p_{i,j} \mid (i,j) \in A \star A\}$ are rationalizable by the mixture model if and only if there exist values x_m, with $0 \leq x_m \leq 1$ for each $\succ_m \in O$, for which*

$$\sum_{\succ_m \in O_{i,j}} x_m = p_{i,j}, \qquad \forall (i,j) \in A \star A. \tag{1}$$

One straightforward way to find out whether a given data set is rationalizable by the mixture model is to check whether there exist nonnegative values x_m that satisfy this system of equalities (1). Similarly, a collection of empirical choice proportions (say, in a human subjects data set from a laboratory experiment) is *rationalizable* if it is consistent with having been generated by choice probabilities that are rationalizable. Determining whether this is the case is a matter of statistical inference subject to the equality constraints (1) on the generating probabilities $\{p_{i,j} \mid (i,j) \in A \star A\}$. Notice that the system of equalities (1) has a variable for every possible preference order of the alternatives, of which there exist $|O| = n!$ many. Even for a moderate number of alternatives, it is computationally prohibitive to solve this system.

Another approach is based on Megiddo's [22] result: A collection $\{p_{i,j} \mid (i,j) \in A \star A\}$ of binary choice probabilities can be viewed as a point in a $n \times (n-1)$-dimensional space. The collection is rationalizable if and only if that point is contained in the *linear ordering polytope*. This polytope (see Section 3.2 for its formulation) can theoretically be described by its facet-defining inequalities, which means that the data set $\{p_{i,j} \mid (i,j) \in A \star A\}$ is rationalizable by the mixture model if and only if the probabilities $p_{i,j}$ satisfy all inequalities defining the linear ordering polytope. However, the number of facet-defining inequalities needed to describe the linear ordering polytope rises very fast with the number of alternatives; a complete description is known for up to 7 alternatives only (see, e.g., Martí and Reinelt [19]). Furthermore, the problem of establishing whether any facet-defining inequalities are violated is NP-COMPLETE for several known classes of inequalities. Here, we circumvent the need to solve a huge system of equalities (1), or to list and check all facet defining inequalities, by moving to a different perspective: column generation.

3 Column Generation

In this section, we describe an algorithm based on column generation to detect whether a given data set can be rationalized by the mixture model. Column generation is a technique dating

back to Gilmore and Gomory [11] who used it to solve cutting stock problems. The advantage of using column generation is that we do not have to consider all of the variables at once; instead, we repeatedly solve a linear program of limited size (the so-called *restricted master*), and we solve a so-called *pricing problem* after each iteration to either establish optimality of the solution found, or to identify new variable(s) to be added to the restricted master. Being able to solve this pricing problem efficiently is key to developing an attractive column generation method. We refer to Chvátal [5] and Lübbecke and Desrosiers [17] for a more detailed description of column generation. In Sections 3.1, 3.2 and 3.3, we look at the setting described previously. In Section 3.4, we briefly show how to adapt the algorithm to different choice settings and decision rules.

3.1 A Linear Programming Formulation

We rewrite the system of equalities given in Definition 1 as a linear programming problem in the following fashion.

$$\begin{aligned}
&\text{Minimize} && z, && && (2)\\
&\text{subject to} && && &&\\
& && \sum_{\succ_m \in O_{ij}} x_m + z \geq p_{i,j}, && \forall (i,j) \in A \star A, && (3)\\
& && \sum_{\succ_m \in O} x_m \leq 1, && && (4)\\
& && x_m, z \geq 0, && \forall \succ_m \in O. && (5)
\end{aligned}$$

Fact 1. *The optimal solution value of (2)-(5) is equal to 0 if and only if nonnegative numbers x_m (hence, preferences $\succ_m \in O$) exist that are a feasible solution to the system of equalities (1).*

Proof. This can be checked as follows. Suppose we have a solution to (3)-(5) with $z = 0$. Consider the following expression for some distinct $i, j \in A$:

$$1 \geq \sum_{\succ_m \in O} x_m = \sum_{\succ_m \in O_{i,j}} x_m + \sum_{\succ_m \in O_{j,i}} x_m \geq p_{i,j} + p_{j,i} = 1. \tag{6}$$

The first inequality follows from (4), and the first equality follows from the fact that the set O can be partitioned into orders where i comes before j and orders where j comes before i. The second inequality follows from (3). The final equality follows from the definition of choice probabilities. Thus, expression (6) is valid, and hence we must have $\sum_{\succ_m \in O_{i,j}} x_m = p_{i,j}$ for each $(i,j) \in A \star A$. This implies that the values x_m are a solution to (1).

Conversely, suppose there exist nonnegative numbers x_m (hence, preferences $\succ_m \in O$) satisfying the system of equalities (1). The values of x_m can then be put into the linear programming problem. Since, for each $(i,j) \in A \star A$, we have $\sum_{\succ_m \in O_{i,j}} x_m = p_{i,j}$ and $\sum_{\succ_m \in O} x_m = 1$, it follows that constraints (3)-(5) are met with $z = 0$. □

Thus, Fact 1 tells us that we can determine whether a given data set is rationalizable by the mixture model by solving the minimization problem (2)-(5). Using standard terminology, we call that minimization problem the *master* problem.

3.2 The Pricing Problem

When we associate dual variables $y_{i,j}$ to constraints (3), and a dual variable c to constraint (4), the dual of (2)-(5) is as follows:

$$\text{Maximize} \quad \sum_{(i,j)\in A\star A} p_{i,j}y_{i,j} - c, \tag{7}$$

subject to

$$\sum_{(i,j)\in A\star A:\succ_m\in O_{ij}} y_{i,j} - c \leq 0, \qquad \forall \succ_m\in O, \tag{8}$$

$$\sum_{(i,j)\in A\star A} y_{i,j} \leq 1, \tag{9}$$

$$y_{i,j}, c \geq 0, \qquad \forall (i,j) \in A\star A. \tag{10}$$

To determine whether a given collection $\{y_{i,j} \mid (i,j) \in A\star A\}$ and a given value c form a feasible solution to this dual problem, one can solve a so-called *pricing problem.* In this case, since constraints (9) and (10) are easy to check, the pricing problem boils down to establishing whether there exists a preference order $\succ_m\in O$, for which $\sum_{(i,j)\in A\star A:\succ_m\in O_{ij}} y_{i,j} > c$. We can formulate this pricing problem using binary variables $b_{i,j}$ that can be associated with each $(i,j) \in A\star A$, as follows. We define $b_{i,j} = 1$ if and only if alternative i is preferred over alternative j. Let $A\star A\star A$ denote the collection of all ordered triples of distinct elements of A, i.e., $A\star A\star A = \{(i,j,k) \mid i \in A, j \in A, k \in A, \text{ with } i,j,k \text{ distinct}\}$. We can formulate the pricing problem as the following maximization problem:

$$\text{Maximize} \quad \sum_{(i,j)\in A\star A, i\neq j} y_{i,j}b_{i,j}, \tag{11}$$

subject to

$$b_{i,j} + b_{j,i} = 1, \qquad \forall (i,j) \in A\star A, \tag{12}$$

$$b_{i,j} + b_{j,k} + b_{k,i} \leq 2, \qquad \forall (i,j,k) \in A\star A\star A, \tag{13}$$

$$b_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A\star A. \tag{14}$$

This pricing problem is the well known linear ordering problem (see Martí and Reinelt [19]) and the convex hull of all solutions satisfying (12) - (14) is the linear ordering polytope. Any solution of this problem for which the objective value (11) is greater than c, corresponds to a violated inequality (8) of the dual. This violated inequality directly corresponds to a primal variable, which, when added, refines the restricted master problem so as to yield a better solution.

3.3 The method

A high-level description of our method is as follows. Initially, we solve a restricted master problem. The formulation of this restricted master problem uses a subset of the variables used in the master problem (2)-(5). Given a solution to the restricted master, we test whether this solution is optimal for the master problem by solving the pricing problem (11)-(14), yielding a solution $\{b^*_{i,j} \mid (i,j) \in A\star A\}$. If the value of this solution is bounded by c, i.e., if $\sum_{(i,j)\in A\star A} y_{i,j}b^*_{i,j} \leq c$, then the current solution to the restricted master problem is in fact optimal for the master problem. Otherwise, we have identified a violated constraint of type (8), and we add the associated primal variable to the restricted master problem, which we then solve again. As there can be no solution to the master problem with $z < 0$, the column generation algorithm terminates as soon as $z = 0$.

Thus, in this description of the method the value c acts as a threshold. We now show how we can strengthen this threshold by making use of the fact that the rationalizability question is a decision problem. Indeed, we are only interested in detecting whether or not a feasible solution

to the master problem exists in which $z = 0$. This allows us to use a stronger stopping condition, based on the objective value of the solutions to the pricing problem.

To proceed, let us define a quantity P as

$$P = \sum_{(i,j)\in A\star A} y_{i,j} p_{i,j}.$$

Notice that, given the $y_{i,j}$ values and the $p_{i,j}$ values, P is trivial to compute.

Theorem 1. *(i) Given numbers $y_{i,j}$ for all pairs $(i,j) \in A \star A$, there exist nonnegative values x_m satisfying (1) only if there exists a linear order $\succ_m \in O$, such that*

$$\sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} \geq P. \tag{15}$$

(ii) Moreover, we have $P \geq c$.

Proof. We prove (i) by contradiction. Suppose that there exist nonnegative values x_m satisfying (1), while for each $\succ_m \in O$

$$\sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} < P.$$

Then, using that, $\forall \succ_m \in O$, $x_m \geq 0$ and that $\sum_{\succ_m \in O} x_m = 1$, we find

$$\sum_{\succ_m \in O} \left(x_m \sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} \right) < P. \tag{16}$$

The left-hand side of (16) can be written as

$$\sum_{\succ_m \in O} \left(x_m \sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} \right) = \sum_{(i,j)\in A\star A} \sum_{\succ_m \in O_{i,j}} x_m y_{i,j}. \tag{17}$$

Thus, using (16) and (17), we arrive at the following inequality:

$$\sum_{(i,j)\in A\star A} \sum_{\succ_m \in O_{i,j}} y_{i,j} x_m < P. \tag{18}$$

Now, since (1) is satisfied, we have, for each ordered pair $(i,j) \in A \star A$, that

$$\sum_{\succ_m \in O_{i,j}} x_m = p_{i,j}. \tag{19}$$

Multiplying both sides of (19) by $y_{i,j}$ preserves the equality; thus, for each $(i,j) \in A \star A$, we have

$$\sum_{\succ_m \in O_{i,j}} y_{i,j} x_m = y_{i,j} p_{i,j}. \tag{20}$$

Summing over all ordered pairs gives

$$\sum_{(i,j)\in A\star A} \sum_{\succ_m \in O_{i,j}} y_{i,j} x_m = \sum_{(i,j)\in A\star A} y_{i,j} p_{i,j} = P. \tag{21}$$

Clearly, equality (21) contradicts (18), and therefore (i) is proved.

To prove (ii), we observe that the value z of the objective function of the restricted master problem equals the value of the objective function of the dual problem ($\sum_{(i,j)\in A\star A} p_{i,j} y_{i,j} - c$), that is,

$$z = \sum_{(i,j)\in A\star A} p_{i,j} y_{i,j} - c = P - c. \tag{22}$$

Since $z \geq 0$, it follows that $P \geq c$. □

This result allows us to end the column generation algorithm if there is no linear order $\succ_m \in O$ for which $\sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} > P$. In Section 4 we show that using this stronger threshold substantially reduces running time.

Note that if condition (15) does not hold, Theorem 1 in effect states that there is a lower bound $z' > 0$ on the optimal solution z^* of the master problem. In general, the convexity constraint on the x_m variables allows us to derive a lower bound on the master problem in every iteration of the column generation algorithm (see Bazaraa et al. [1], Lübbecke and Desrosiers [17]).

We also note that Theorem 1 can be seen as an application of Farkas' Lemma [9]. Indeed, consider this slight rephrasing of the original system of equalities, in which we make the convexity constraint explicit. Let

$$\sum_{\succ_m \in O_{ij}} x_m = p_{i,j}, \qquad \forall (i,j) \in A \star A, \tag{23}$$

$$\sum_{\succ_m \in O} x_m = 1, \tag{24}$$

$$x_m \geq 0, \qquad \forall \succ_m \in O. \tag{25}$$

Then Farkas' Lemma states that this system has a solution if and only if there does not exist a solution $(f_{i,j}, g)$ to the following system of inequalities.

$$\sum_{(i,j)\in A\star A: i\succ_m j} f_{i,j} + g < 0, \qquad \forall \succ_m \in O, \tag{26}$$

$$\sum_{(i,j)\in A\star A} p_{i,j} f_{i,j} + g \geq 0. \tag{27}$$

However, given a dual solution $\{c, y_{i,j} \mid (i,j) \in A \star A\}$ violating condition (15), i.e., given that no linear order $\succ_m \in O$ can satisfy (15), a solution to (26-27) must exist: Let $f_{i,j} = y_{i,j}$. Then, for each $\succ_m \in O$:

$$\sum_{(i,j)\in A\star A: i\succ_m j} f_{i,j} < \sum_{(i,j)\in A\star A} p_{i,j} f_{i,j}. \tag{28}$$

Given this, there must exist some g, such that

$$\sum_{(i,j)\in A\star A: i\succ_m j} f_{i,j} + g < 0 \leq \sum_{(i,j)\in A\star A} p_{i,j} f_{i,j} + g. \tag{29}$$

We close this section with a pseudo-code version of our algorithm. For details on how to select an initial set of variables (Line 1) specifying the restricted master problem and how exactly to solve the Pricing Problem (Line 6), see Section 4. The value labeled "Threshold" in Line 7 stands for either c or P, that is, the value of the pricing solution below which the algorithm concludes that the data are nonrationalizable (see also Theorem 1).

3.4 Generalization of our approach

The approach we describe in the previous subsections is not restricted to either linear orderings or a binary choice setting. Here we describe two different variations of the model, and we illustrate how modifications of our approach are still valid ways to decide rationalizability.

Indeed, suppose that, instead of choice ratios coming from binary choice, we are given data from a ternary choice situation: Here, writing A^3 to denote the set of (unordered) triples consisting of three distinct alternatives from A, we have numbers $p_{i,\{i,j,k\}}$ (respectively $p_{j,\{i,j,k\}}$, or $p_{k,\{i,j,k\}}$) denoting the probability that alternative i (respectively j, or k) is preferred among the triple $\{i,j,k\} \in A^3$. Then, writing $O_{i,S}$ to denote the set of linear orders in which alternative i is the most preferred alternative of the set S, the following definition applies.

Algorithm 1 Column Generation INPUT: $A, \{p_{i,j} \mid (i,j) \in A \star A\}$

1: Solve Restricted Master Problem
2: **if** $z = 0$ **then**
3: OUTPUT: Yes, rationalizable
4: **else**
5: Update Pricing Problem with values $\{y_{i,j} \mid (i,j) \in A \star A\}$
6: Solve Pricing Problem
7: **if** Value Pricing Solution $\leq$ Threshold **then**
8: OUTPUT: No, not rationalizable
9: **else**
10: Add to Restricted Master Problem the variable(s) corresponding to linear order(s) found in Line 6
11: GOTO Line 1
12: **end if**
13: **end if**

Definition 2. *Choice probabilities $\{p_{i,S} \mid S \in A^3, i \in S\}$ are rationalizable by the mixture model if and only if there exist values x_m, with $0 \leq x_m \leq 1$ for each $\succ_m \in O$, for which*

$$\sum_{\succ_m \in O_{i,S}} x_m = p_{i,S}, \qquad \forall S \in A^3, i \in S. \tag{30}$$

Next, the analog of the model (2)-(5) for the setting with ternary choice becomes the following model.

$$\text{Minimize} \qquad z, \tag{31}$$

subject to

$$\sum_{\succ_m \in O_{i,S}} x_m + z \geq p_{i,S}, \qquad \forall S \in A^3, i \in S, \tag{32}$$

$$\sum_{\succ_m \in O} x_m \leq 1, \tag{33}$$

$$x_m, z \geq 0, \qquad \forall \succ_m \in O. \tag{34}$$

Following the arguments in Section 3.2, we arrive at the following pricing problem: We need to determine whether there exists a preference order $\succ_m \in O$, for which

$$\sum_{S \in A^3} \sum_{i \in S : \succ_m \in O_{i,S}} y_{i,S} > c. \tag{35}$$

The $y_{i,S}$ represent the dual variables corresponding to (32).

When formulating this pricing problem as an integer program, using - next to the already defined binary variables $b_{i,j}$ - binary variables $r_{i,S}$ equalling 1 if and only if i is the most preferred alternative in S, we arrive at the following model.

$$\text{Maximize} \qquad \sum_{S \in A^3} \sum_{i \in S : \succ_m \in O_{i,S}} y_{i,S} r_{i,S}, \tag{36}$$

subject to

$$b_{i,j} + b_{j,i} = 1, \qquad \forall (i,j) \in A \star A, \tag{37}$$

$$b_{i,j} + b_{j,k} + b_{k,i} \leq 2, \qquad \forall (i,j,k) \in A \star A \star A, \tag{38}$$

$$r_{i,S} \leq b_{i,j}, \qquad \forall S \in A^3, \forall i,j \in S, i \neq j, \tag{39}$$

$$b_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A \star A, \tag{40}$$

$$r_{i,S} \in \{0,1\}, \qquad \forall S \in A^3, \forall i \in S. \tag{41}$$

Notice that one can readily generalize this model beyond ternary choice and adapt it to settings where the sets S contain more than three choice alternatives.

Another generalization arises when choice is not forced. In other words, when confronted with a pair of alternatives, a respondent is allowed to respond by not selecting a favorite alternative. We consider this choice to mean that the respondent is indifferent between the respective alternatives. Thus, for every ordered pair (i, j) of alternatives we are given a nonnegative number $p_{i,j}$; now however, in contrast to Section 2, we only know that $p_{i,j} + p_{j,i} \leq 1$. In fact, $1 - p_{i,j} - p_{j,i}$ represents the probability that a respondent is indifferent between alternatives i and j.

To accommodate indifference between alternatives, we consider weak orders [26]. A weak order can be seen as a partition of the set of alternatives into ordered equivalence classes. We use the index m to indicate a particular weak order. If, for the weak order $\succ_m$, the alternative $i \in A$ is preferred over the alternative $j \in A$, we write $i \succ_m j$. If, for the weak order $\succ_m$, the alternatives i and j are in the same equivalence class, then we write $i \sim_m j$. We denote the set of all weak orders as W. We further consider the subsets $W_{i,j}$ denoting the set of all weak preference orders in which alternative i is preferred over alternative j, and W_{i-j} denoting the set of all weak preference orders in which alternatives i and j are in the same equivalence class $(i, j \in A, i \neq j)$. We modify Definition 1 as follows.

Definition 3. *Choice probabilities $\{p_{i,j} \mid (i,j) \in A \star A\}$ are rationalizable by a mixture model of weak orders if and only if there exist values x_m, with $0 \leq x_m \leq 1$ for each $\succ_m \in W$, for which*

$$\sum_{\succ_m \in W_{i,j}} x_m = p_{i,j}, \qquad \forall (i,j) \in A \star A. \tag{42}$$

$$\sum_{\succ_m \in W_{i-j}} x_m = 1 - p_{i,j} - p_{j,i}, \qquad \forall \{i,j\} \in A^2. \tag{43}$$

The analog of the model (2)-(5) for the setting with indifference becomes as follows.

$$\text{Minimize} \qquad z, \tag{44}$$

$$\text{subject to}$$

$$\sum_{\succ_m \in W_{i,j}} x_m + z \geq p_{i,j}, \qquad \forall (i,j) \in A \star A, \tag{45}$$

$$\sum_{\succ_m \in W_{i-j}} x_m + z \geq p_{i,j}, \qquad \forall \{i,j\} \in A^2, \tag{46}$$

$$\sum_{\succ_m \in W} x_m \leq 1, \tag{47}$$

$$x_m, z \geq 0, \qquad \forall \succ_m \in W. \tag{48}$$

We arrive at the following pricing problem: Writing $y_{i,j}$ for the dual variables corresponding to inequalities (45), and $u_{i,j}$ for the dual variables corresponding to inequalities (46), we need to determine whether there exists a weak preference order $\succ_m \in W$ for which

$$\sum_{i,j \in A \star A : i \succ_m j} y_{i,j} + \sum_{i,j \in A^2 : i \sim_m j} u_{i,j} > c. \tag{49}$$

We can reformulate this pricing problem using binary variables $b_{i,j}$ that equal 1 when alternative i is preferred over alternative j, and equal 0 when alternatives i and j are in the same equivalence

class.

$$\text{Maximize} \quad \sum_{(i,j)\in A\star A, i\neq j} (y_{i,j} - u_{i,j})b_{i,j}, \tag{50}$$

subject to

$$b_{i,j} + b_{j,i} \leq 1, \qquad \forall (i,j) \in A \star A, \tag{51}$$

$$b_{i,j} + b_{j,k} + b_{k,i} \leq 2, \qquad \forall (i,j,k) \in A \star A \star A, \tag{52}$$

$$b_{j,k} - b_{j,i} \leq b_{i,k}, \qquad \forall (i,j,k) \in A \star A \star A, \tag{53}$$

$$b_{i,j} - b_{k,j} \leq b_{i,k}, \qquad \forall (i,j,k) \in A \star A \star A, \tag{54}$$

$$b_{i,j} \in \{0,1\}, \qquad \forall (i,j) \in A \star A. \tag{55}$$

Notice that inequalities (51) (in contrast to equalities (12)) allow alternatives i and j to be in the same equivalence class. In addition inequalities (53) and (54) imply that if alternatives i and j, as well as alternatives i and k, are in the same equivalence class, then alternatives j and k should also be in the same equivalence class. Hence, (51)-(55) gives a weak order.

These examples illustrate the versatility of the column generation approach. Any mixture model, for which the underlying decision rule can be modeled to obtain a pricing problem falls under this framework. Obtaining efficient solution methods may require tailor-made methods to solve the pricing problem.

4 Implementation

In this section we discuss the implementation of the column generation algorithm. On the one hand, we can run Algorithm 1 by making use of integer programming solvers to solve the pricing problem. However, on the other hand, fast heuristics may already give solutions with a value exceeding the threshold. We describe these heuristic algorithms in Subsection 4.1. Subsection 4.2 then contains descriptions of our data sets. Finally, Subsection 4.3 gives results on computation times for the various algorithms and data sets.

4.1 Heuristic Algorithms

As the linear ordering problem is a well-known and well-studied NP-HARD problem, there is an extensive literature on heuristic algorithms to solve it. In this section, we describe our implementation based on best insertion constructive and local search algorithms (Laguna et al. [16]). The literature has observed that these perform well when compared to other simple heuristics (see Martí and Reinelt [19]). We use multi-start procedures that vary the order in which we add alternatives. In this way, we have multiple solutions to compare against each other. These multiple solutions can be used to either pick the best solution, or to identify multiple variables to add to the restricted master in the column generation. Furthermore, we describe an algorithm for the pricing problem that, under certain circumstances, adjusts the given dual solution in order to find 'promising' linear orders (that is, orders whose variables we expect to be positive in the solution to the master problem).

4.1.1 Best Insertion Heuristics

We describe a constructive heuristic called Best Insertion (Algorithm 2); and a local search method (Algorithm 3) based on a 'move' neighbourhood (see Martí and Reinelt [19]). The Best Insertion algorithm creates an initial ranking of the alternatives by iteratively placing alternatives in an order over a (sub)set of the alternatives. In the local search method, the position of alternatives in this order can be changed by local moves. Initially, we consider the set A of all alternatives. For every $(i,j) \in A \star A$, the value of placing i before j is given by $y_{i,j}$. Let $\langle a_1, a_2, \ldots, a_k \rangle$, with $1 \leq k \leq n$ denote a linear order of k many elements in A.

Algorithm 2 Best Insertion — INPUT: A, $a_1 \in A$, $\{y_{i,j} \mid (i,j) \in A \star A\}$

Set $A := A \backslash \{a_1\}$
Create an order $\langle a_1 \rangle$
Set $k := 1$
while $A \neq \varnothing$ **do**
  Let $\langle a_1, a_2, \ldots, a_k \rangle$ denote the current order
  Choose an alternative $i \in A$.
  For each $t = 1, \ldots, k+1$, compute $q_t = \sum_{j=1}^{t-1} y_{a_j,i} + \sum_{j=t}^{k} y_{i,a_j}$
  Let $r = \arg\max_{t=1,\ldots,k+1} q_t$
  Set $j := k+1$
  while $j > r+1$ **do**
    $a_j := a_{j-1}$
    $j := j-1$
  end while
  $a_r := i$
  $A := A \backslash \{i\}$
end while
OUTPUT: Linear order a

For the local search algorithm, let $v(a)$ be the objective value associated with a linear order $\langle a_1, a_2, \ldots, a_n \rangle$ in the pricing problem, i.e., $v(a) = \sum_{i<j} y_{i,j}$. Furthermore, let $v(a, \ell, m)$, be the value of the order that results when the alternative in position ℓ is moved to position m.

Algorithm 3 Local Search — INPUT: $a = \langle a_1, a_2, \ldots, a_n \rangle$, $\{y_{i,j} \mid (i,j) \in A \star A\}$

Set $i := 1$
while $i < n+1$ **do**
  if $\max_{j=1,\ldots,n+1} v(a,i,j) > v(a)$ **then**
    Set $a := \langle \ldots, a_{j-1}, a_i, a_j, \ldots \rangle$
    Set $i := 1$
  else
    Set $i := i+1$
  end if
end while
OUTPUT: Linear order a

In this local search heuristic, we define the *neighborhood* of an order as the collection of all orders that can be constructed from the current order by moving a single alternative to a different position. For a given alternative, we evaluate all possible such moves. If the best possible move for this alternative improves the objective value, then we implement this move and we update the order. The algorithm terminates if there are no more improvements possible through moving a single alternative. Algorithm 3 gives the full pseudo-code for the local search.

In Algorithm 4, we show how we combine the algorithms described so far. We denote the best order found so far by $\bar{a}$. This implementation combines Best Insertion with Local Search to quickly find linear orders that are good solutions to the pricing problem. Since the outcome of the constructive heuristic depends strongly on the order in which alternatives are added to the linear order, we use a multi-start procedure. For each alternative $i \in A$, we run the algorithm once, inserting i first. From these multiple runs, we save the best solution to the pricing problem, and if this solution can be used to add a variable to the restricted master problem, we do so. If the objective value found through the heuristics is not strictly positive, then we have not found any variables to add to the master problem. However, this does not mean that no such variables exist. Therefore, as a back-up, we use an exact solver that either finds a new variable, if one exists, or provides us with proof that such a variable does not exist. In this way, we are still guaranteed a

Algorithm 4 Pricing Problem (Single Solution) INPUT: A, $\{y_{i,j} \mid (i,j) \in A \star A\}$

1: Set $V := -\infty$
2: **for** $i \in A$ **do**
3: Run Best Insertion (Algorithm 2), starting with alternative i, to output linear order a
4: Run Local Search (Algorithm 3) with starting solution a, to output updated a
5: If $v(a) > V$, set $V := v(a)$ AND $\bar{a} := a$
6: **end for**
7: **if** $v(\bar{a}) \leq 0$ **then**
8: Solve the pricing problem exactly
9: **else**
10: OUTPUT: Linear order $\bar{a}$
11: **end if**

correct test of the mixture model.

To further speed up the Column Generation algorithm, we look for multiple solutions to our pricing problem. By adding additional variables in a single iteration, we hope to get larger improvements in the objective function value of the restricted master problem. In our implementation, we keep using a multi-start Best Insertion heuristic that saves, in a set B, every solution that provides a new improving variable.

Algorithm 5 Pricing Problem (Multiple Solutions) INPUT: A, $\{y_{i,j} \mid (i,j) \in A \star A\}$

1: Set $B := \varnothing$
2: **for all** $i \in A$ **do**
3: Run Best Insertion (Algorithm 2), starting with alternative i, to output linear order a
4: Run Local Search (Algorithm 3), with starting solution a
5: If $v(a) > 0$, set $B := B \cup \{a\}$
6: **end for**
7: **if** $B = \varnothing$ **then**
8: Solve the pricing problem exactly
9: **else**
10: OUTPUT: Collection B of linear orders
11: **end if**

4.1.2 Adjusting pricing solutions

As a final addition to the column generation algorithm in our implementation, we propose a way to adjust a solution to the pricing problem, so that it matches the data as closely as possible. More specifically, the adjustment we propose is as follows. Suppose that, as a solution to the pricing problem, we have found a linear order with $i \succ j$ that we can add to the restricted master problem as an improving column. If $y_{i,j} = y_{j,i} = 0$, then a linear order with $j \succ i$ has the same objective value in the pricing problem. Now suppose that $p_{j,i} = 1$. In that case, it follows that no order with $i \succ j$ can be used in the final solution to the master problem, and we should add an order with $j \succ i$ instead. Similar reasoning applies when $p_{j,i} > 0.5$. Here, we cannot rule out that a variable with $i \succ j$ will be used. Even so, it is plausible that the eventual solution allocates more weight to variables corresponding to orders with $j \succ i$. We therefore add the steps outlined in Algorithm 6 below to the end of our heuristic pricing algorithms (i.e., we insert Algorithm 6 between line 4 and 5 in Algorithm 4, and between line 4 and line 5 in Algorithm 5). In effect, we solve a new linear ordering problem with values $\overline{y_{i,j}}$. We set these values in such a way that the preference order returned by this second problem has at least the same objective value as the original pricing problem when using the original $y_{i,j}$ values in the objective. For every pair $(i,j) \in A \star A$, for which $y_{i,j} > 0$ and $b^*_{i,j} = 1$ we set $\overline{y_{i,j}}$ equal to an arbitrarily high number (1000 in our application). This guarantees that the new solution still satisfies $i \succ j$. For every pair

$(i,j) \in A \star A$, for which $y_{i,j} = y_{j,i} = 0$, we set $\overline{y_{i,j}} := p_{i,j}$ and $\overline{y_{j,i}} := p_{j,i}$. We set all other values of $\overline{y_{i,j}}$ equal to 0.

Algorithm 6 Pricing Problem (Adjusted) INPUT: A, $a = \langle a_1, a_2, \ldots, a_n \rangle$, $\{y_{i,j} \mid (i,j) \in A \star A\}$

1: **for all** $(i,j) \in A \star A$ **do**
2: Set $\overline{y_{i,j}} := 0$
3: If $i \succ j$ in a and $y_{i,j} > 0$, set $\overline{y_{i,j}} := 1000$
4: If $y_{i,j} = y_{j,i} = 0$, set $\overline{y_{i,j}} := p_{i,j}$
5: **end for**
6: Run Local Search (Algorithm 3), with starting solution a

4.2 Data Sets

We generated four distinct classes of binary (forced) choice data sets[1], each with $n = 20$. Two classes of data sets, called *Inside Easy* (IE) and *Inside Hard* (IH), satisfy the mixture model, whereas the other two classes of data sets, called *Outside Easy* (OE) and *Outside Hard* (OH), violate it.

For IE and IH, we first randomly generated t linear orders over the 20 alternatives. For each alternative, we then randomly drew a number between 0 and 100. Next, we ranked all alternatives by the size of the generated number in order to obtain linear orders (in case of a tie, the lower indexed alternative came first). For Inside Easy, we used $t = 20$, whereas, for Inside Hard, we used $t = 5$. Next, we drew $t-1$ random numbers $(q_m, m = 1, 2, \ldots, t-1)$, between zero and one, which we then ranked from small to large. This yielded t intervals $[0, q_1], [q_1, q_2], \ldots, [q_{t-1}, 1]$, that defined t numbers $q_i - q_{i-1}$ (with $q_0 = 0$ and $q_t = 1$). We let each of these t numbers correspond to a different generated linear order, i.e., we set $x_m = q_m - q_{m-1}$ for $m = 1, \ldots, t$. We then set the $p_{i,j}$ values as $p_{i,j} = \sum_{\succ_m \in O_{ij}}^{t} x_m$.

We generated OE and OH in such a way that they were unlikely to satisfy the mixture model. In the case of Outside Easy, for every pair of distinct alternatives $i, j \in A$, we drew a number $p_{i,j}$ from a uniform distribution between zero and one and we set $p_{j,i} = 1 - p_{i,j}$. While this generation process did not guarantee nonrationalizability, it turned out that all of the data sets that we obtained in this fashion did indeed violate the mixture model. This is due to the restrictive nature of this model. Monte Carlo simulation shows that only about 5% of data sets containing 5 alternatives generated in this manner satisfy the mixture model, and that this percentage decreases with an increasing number of alternatives [24]. For Outside Hard, we drew initial $p_{i,j}$ values using the same procedure. We used these numbers as the input to an optimization problem that minimized the changes in the $p_{i,j}$ values, under the constraint that for every triple of distinct alternatives, $(i,j,k) \in A \star A \star A$, the inequality $p_{i,j} + p_{j,k} + p_{k,i} \leq 2$ held. This basic inequality, which is called the *triangle inequality*, is well-known to be facet defining for the linear ordering polytope. The resulting data sets were generally much closer to satisfying the mixture model than $p_{i,j}$ values drawn from a uniform distribution. However, all data sets we obtained using this procedure were still nonrationalizable.

4.3 Computational Results

In this section, we consider computational experiments. In all cases, we used a PC with an Intel i5 quad core 3.4 GHz processor and 4 GB RAM. We used CPLEX 12.4 for finding the exact solutions to the pricing problem (Line 8 in Algorithms 4 and 5) and for solving the restricted master problems (Line 1 in Algorithm 1).

In Table 1, we compare the computation times needed by Column Generation (Algorithm 1) depending upon different ways in which we solved the pricing problem in Line 6. We distinguish three cases: (i) Using an exact solver (here CPLEX; we display the results in the upper subtable of

[1]The data sets are publicly available at `http://hdl.handle.net/2268/207262`

Table 1), (ii) using Algorithm 4 that generates a single order (we display the results in the middle subtable of Table 1), or (iii) using Algorithm 5 that potentially generates multiple orders (we display the results in the lower subtable of Table 1). The first column, entitled "Data Set Class", describes the particular class of instances; recall that a particular class contains 20 instances. Columns 2, 3, 4 and 5 give average times in seconds, while Columns 6 and 7 give average numbers of iterations, and Column 8 gives, if applicable, the average number of orders added. More precisely, the second column, entitled "Decide Rationalizability", gives the total time spent by Column Generation on determining whether the instance is rationalizable or not. We summarize the most important sources for that time in the following three columns. "Pricing P. Exact" gives the total time spent by CPLEX on solving the pricing problem exactly. "Restricted Master" gives the total time spent by CPLEX on solving the restricted master problems. "Pricing Problem Single Solution" (respectively, "Multiple Solutions") is the total time spent on solving the pricing problem heuristically using Algorithm 4 (respectively, Algorithm 5). The sixth column gives the total number of iterations in which the pricing problem was solved, whereas the seventh column reports how many of those iterations used CPLEX. The last column shows the total number of linear orders generated by Algorithm 5 (when using Algorithm 4, the number of iterations equals the number of orders generated, and hence we do not explicitly repeat this information in this column).

	Column Generation with exact solver (Algorithm 1 with CPLEX in Line 6)				
Data Set Class	Average Time per Instance in Seconds			Total Number of	
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Iterations Pricing P.	
Inside Easy	12,05	9,73	2,00	244,2	
Inside Hard	1577,57	1524,14	41,13	1866,8	
Outside Easy	2,01	1,85	0,10	85,7	
Outside Hard[2]	>3600	>3600	>3600	-	

	Column Generation with Pricing Problem - Single Solution (Algorithm 1 with Algorithm 4 in Line 6)					
Data Set Class	Average Time per Instance in Seconds				Total Number of	
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Pricing Problem Single Solution	Iterations Pricing P.	Iterations Pricing P.-Exact
Inside Easy	2,35	0,00	1,86	0,27	261,4	0,0
Inside Hard	100,40	20,70	60,68	1,76	2238,1	11,2
Outside Easy	0,17	0,02	0,09	0,03	76,9	1,0
Outside Hard	1493,56	1448,15	32,42	1,36	1994,7	120,5

	Column Generation with Pricing Problem - Multiple Solutions (Algorithm 1 with Algorithm 5 in Line 6)						
Data Set Class	Average Time per Instance in Seconds				Total Number of		
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Pricing Problem Multiple Solutions	Iterations Pricing P.	Iterations Pricing P.-Exact	Orders Generated Pricing P.-Multiple
Inside Easy	1,15	0,00	0,82	0,02	30,2	0,0	556,1
Inside Hard	176,21	16,30	109,23	0,35	417,6	10,3	4405,1
Outside Easy	0,19	0,02	0,07	0,01	15,2	1,0	295,6
Outside Hard	1446,28	1393,68	32,15	0,34	499,1	124,4	3096,2

Table 1: Computational results for Column Generation with CPLEX, Algorithm 4, or Algorithm 5 for solving the pricing problem.

We note that the computation times decrease substantially when we use heuristic methods for the pricing problem. Generally, these methods are able to find improving columns in most iterations of the pricing problem, allowing us to skip the computationally expensive exact pricing problems. In fact, for many satisfying data sets, in particular all instances of "Inside Easy", the mixture model test works without having to run any exact tests. For the data sets with violations, at least one exact test is necessary for a guarantee that no improving columns exist. For the Outside Easy data sets, no extra exact pricing problems were necessary. For the Outside Hard data sets, however, we spend most of the run time on solving exact pricing problems. Furthermore, we note that for these instances running Algorithm 4 versus running Algorithm 5 does not lead to large differences in computation time. However, it is also true that while overall computation

[2]Computation time exceeded 1 hour for all 20 data sets.

times are similar, that time splits differently over different parts of the algorithm. Indeed, when adding multiple columns in each pricing iteration (i.e., when running Algorithm 5), the number of pricing iterations that are necessary decreases substantially. However, the total number of added columns is much larger, which, in turn, increases the time needed in updating and running the master problem. Notice that the computation times for pricing and master problems do not add up to the total computation time. The remaining time consists of miscellaneous tasks such as updating the master problem. In the case where all improving columns are added, this can become an important factor in computation times.

In Table 2, we consider how using Algorithm 6 (which adjusts the pricing solutions as described in Section 4.1.2), impacted the results. The results in Table 2 show that adjusting the pricing solutions had a positive impact on average computation times for all classes of data sets. This reduction in computation time was, depending on the basic algorithm, due to two effects. First, when using Algorithm 4, the adjustment of pricing solutions (i.e., also using Algorithm 6) led to a large decrease in the number of iterations that the column generation algorithm required, as witnessed by comparing the corresponding numbers reported in Column 6; this occurred for all types of instances. Second, when using Algorithm 5, the improvement due to adjusting the pricing solutions was mainly due to the fact that a smaller number of variables wsa being added in each iteration; this follows from comparing the corresponding numbers in the last column. In both cases, total computation time goes down on average. However, while the adjustment sped up computation times on average, we found significant variance between instances. As an example, when using Algorithm 4, computation time for one Outside Hard instance increased from 963s to 1108s when using the adjustment, while computation time for another instance decreased from 1879s to 1593s. We close this discussion by pointing out that when computation times increased for an instance, this was always due to an increase in the number of exact pricing iterations and overall longer integer program computation times.

	Column Generation with Adjusted Pricing Problem - Single Solution (Algorithm 1 with Adjusted Algorithm 4 in Line 6)					
Data Set Class	Average Time per Instance in Seconds				Total Number of	
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Pricing Problem Single Solution	Iterations Pricing P.	Iterations Pricing P.-Exact
Inside Easy	1,66	0,00	1,30	0,22	190,10	0,00
Inside Hard	60,91	10,69	37,29	2,57	1776,70	5,60
Outside Easy	0,14	0,02	0,02	0,10	26,90	1,00
Outside Hard	1447,34	1413,67	23,48	1,86	1677,70	119,90

	Column Generation with Adjusted Pricing Problem - Multiple Solutions (Algorithm 1 with Adjusted Algorithm 5 in Line 6)						
Data Set Class	Average Time per Instance in Seconds				Total Number of		
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Pricing Problem Multiple Solutions	Iterations Pricing P.	Iterations Pricing P.-Exact	Orders Generated Pricing P.-Multiple
Inside Easy	1,08	0,00	0,64	0,025	49,30	0,00	414,50
Inside Hard	133,85	26,44	72,57	0,68	444,30	16,30	3666,40
Outside Easy	0,07	0,02	0,01	0,03	7,80	1,00	83,40
Outside Hard	1408,80	1377,87	18,03	0,46	470,10	122,00	2515,10

Table 2: Computational results including adjustments to the pricing solutions.

Finally, we investigate the effect of using the stronger stopping condition as discussed in Section 3.3. Table 3 reports results for the Outside Hard data sets; notice that a row in Table 3 no longer corresponds to a class of instances, but instead to a specification of the main algorithm Column Generation. It is clear from the table that using this more stringent stopping has a dramatic impact on the computation times. The algorithm now quickly identifies data sets that cannot be consistent with the mixture model, cutting average computation times from more than 20 minutes to less than half a minute. In each tested instance of the Outside Hard class, the strong stopping condition terminates the column generation in the first iteration that uses an exact method for the pricing problem. The impact of using this strong stopping conditions exceeds the impact of all other choices made when specifying the algorithm.

	Column Generation with/without Adjusted Pricing Problem, with/without Stronger Stopping Condition						
Pricing Problem	Average Time per Instance in Seconds				Total Number of		
	Decide Rationalizability	Pricing P. Exact	Restricted Master	Pricing Problem Multiple Solutions	Iterations Pricing P.	Iterations Pricing P.-Exact	Orders Generated Pricing P.-Multiple
Single Solution Original	1493,56	1448,15	32,42	1,36	1994,70	120,50	
Adjusted	1447,34	1413,67	23,48	1,86	1677,70	119,90	
Adjusted & Strong	25,34	4,31	15,16	1,54	1239,90	1,00	
Multiple Solution Original	1446,28	1393,68	32,15	0,34	499,10	124,40	3096,20
Adjusted	1408,80	1377,87	18,03	0,46	470,10	122,00	2515,10
Adjusted & Strong	15,78	3,69	6,07	0,24	165,30	1,00	1971,30

Table 3: Impact of adjustments to the pricing solutions and/or stopping condition (Outside Hard only).

5 Testing Many Similar data sets: Bayes Factor Calculation

In this section, we discuss an application of the column generation algorithm to a statistical problem described by Cavagnaro and Davis-Stober [3]. Those authors used the Bayes factor (Klugkist and Hoijtink [15]) for statistical model evaluation, model selection, and model competition on data from human subject experiments in the laboratory. The calculation of the Bayes factor requires evaluating a large number of data sets against the conditions of the probabilistic models of choice. Efficient algorithms for these tests are essential: For example, [23] expended more than 24000 CPU hours, nearly all of them to compute Bayes factors, for their statistical analyses. We show that our column generation approach has additional advantages in this application, as it can leverage information from testing one data set to speed up the tests for the following data sets.

The Bayes factor is defined as the ratio of the marginal likelihoods of two models. First, given observed behaviour, a *posterior distribution* can be calculated for the mixture model in question. This distribution represents how likely specific choice probabilities are to generate the observed data. From this posterior distribution, we sample choice probabilities and test whether these satisfy the mixture model. The percentage of such sampled data sets that satisfy the mixture model (out of the total number of tested samples), provides an approximation of the posterior probability that, given the observed choices, the decision maker satisfies the mixture model. To calculate the Bayes Factor between the mixture model and an unconstrained model that places no restrictions on choice behavior, we divide this posterior probability by the percentage of samples from a prior distribution that are consistent with the model. In line with [3], we take the prior distribution of a $p_{i,j}$ value to be a uniform distribution between 0 and 1. The percentage of samples consistent with the mixture model is then equal to the volume of the linear ordering polytope compared to the unit hypercube. We are mainly interested in computing the posterior probability. The posterior distributions of the $p_{i,j}$ values are given by Beta distributions. In particular, let $q_{i,j}$ be the rate at which i is observed chosen in a choice between i and j. Then, for every pair of alternatives $(i,j) \in A \star A$, the distribution of $p_{i,j}$ is given by $\text{Beta}(q_{i,j}+1, q_{j,i}+1)$. Given a sampled $p_{i,j}$ value for every pair, we have a data set for which the mixture model can be tested. To keep track of the fact that we generate these data sets from the posterior distribution, we call them *synthetic* data sets. To estimate the posterior probability as closely as possible, large numbers of these synthetic data sets must be tested for consistency with the mixture model. However, since all data sets are sampled from the same distribution, these data sets are usually quite similar. In this section, we consider ways in which our column generation procedure can exploit these similarities to quickly test many synthetic data sets. In Subsection 5.1 we look at how re-using the columns generated in one test as a starting set for new data sets speeds up these tests. In Subsection 5.2 we show that the objective function of the final pricing problem of a rejected data set provides inequalities that are also necessary conditions for rationalizability by the mixture model. We can use these inequalities to test more quickly whether data sets violate the mixture model.

5.1 Starting Sets

As all data sets generated from the same distribution are fairly similar, it is likely that the optimal solutions to the linear programs (2-5) for these data sets use many of the same variables. By

testing the points sequentially and using the variables generated while testing previous data sets as a starting set for new data sets, we attempt to minimize the number of pricing problems we need to solve.

We illustrate this by drawing 10000 synthetic data sets from the posterior distribution from a laboratory study with 8 choice alternatives. Table 4 shows the cumulative number of variables generated in the column generation algorithm over these 10000 synthetic data sets. In this test, we used the Column Generation algorithm using Algorithm 4 (i.e., generating a single linear order in each iteration) in conjunction with Algorithm 6 (i.e., using the adjusted pricing procedure. Table 4 shows how many variables were generated to test a number of synthetic data sets. For example, to test the first 100 data sets, 126 variables were generated by the column generation algorithm. The table clearly shows that tests of the first data sets require a larger number of variables to be generated. Evaluating later data sets could mostly be done using already generated variables. In total, we only needed to generate 540 different variables to test all 10000 data sets. In the first 1% of the data sets that we tested, we generated more than 10% of these variables; in the first 10% of the data sets, we generated more than 50% of the variables. The number of variables generated quickly tailed off, as the starting sets for later tests were generally sufficient to prove violation of the mixture model.

# Data sets tested	1	2	5	10	100	1000	2500	5000	10000
# Variables generated	14	25	36	60	126	278	382	448	540

Table 4: Cumulative number of variables generated.

5.2 Valid Inequality Pool

Using the starting sets as described in the previous subsection offers substantial speed-ups for testing many similar data sets. However, one issue that remains is that, for each nonrationalizable data set, to prove that we have reached the optimal solution to the master problem, we must solve an exact pricing problem. Since exact pricing problems contribute heavily to overall computation times, we wish to avoid this if possible. This is the motivation behind the following theorem.

Observation 1. *Suppose there exist numbers $y_{i,j}$ for all $(i,j) \in A \star A$, and a number c, such that there does not exist a linear order $\succ_m \in O$ for which $\sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} > c$. Then there exists no data set with numbers $p_{i,j}$ for all $(i,j) \in A \star A$, that rationalizes the mixture model and for which $\sum_{(i,j)\in A\star A} y_{i,j}p_{i,j} > c$.*

Proof. The proof is analogous to the proof for Theorem 1. □

The intuition behind this theorem is as follows: if we have a hyperplane, defined by the $y_{i,j}$ values and c, and there is no extreme point of the polytope beyond this hyperplane (i.e. no linear order $\succ_m \in O$ for which $\sum_{(i,j)\in A\star A: i\succ_m j} y_{i,j} > c$), then there does not exist any point in the polytope beyond the hyperplane ($\sum_{(i,j)\in A\star A} y_{i,j}p_{i,j} > c$). In other words, it is a separating hyperplane. Each time we solve a pricing problem, we encounter such a hyperplane. Furthermore, the value of the optimal solution of the pricing problem provides the value c. Each time the pricing problem is solved exactly, we thus obtain an inequality that all rationalizable data sets must satisfy. We choose to save these inequalities in the iteration in which the column generation terminates. This means that, before using the column generation algorithm on additional data sets, we can first test whether these data sets violate any of the inequalities we have identified so far. If we find violations, then we conclude that the data set does not satisfy the mixture model and that no further test is necessary.

For some problems, such as the linear ordering problem, there already exists a wealth of information on valid and facet-defining inequalities. We can use these known inequalities in the same way as valid inequalities identified during the column generation. This can further reduce computation

time as identifying valid inequalities requires solving computationally expensive exact pricing problems. In fact, nearly all of the valid inequalities we identify in our experiment are of the well-known class of triangle inequalities. However, to show our approach also works for problems for which no or few classes of valid inequalities are already known, we will only rely on valid inequalities identified throughout the column generation.

5.3 Computational Results

We present results from a computational study for Bayes factor generation using posterior distributions from a laboratory study. The laboratory study used eight choice alternatives and collected separate sets of responses from 32 human participants. For each of the resulting 32 posterior distributions, we generated 10000 synthetic data sets and tested whether they were rationalizable. The number of rationalizable synthetic data sets, out of 10000, varied between 0 and 235. In these computational tests, we used Column Generation with Algorithm 4, Algorithm 6, and the strong stopping condition.

First, let us note that in these tests, the individual rationalizability tests are easier than those in the previous section, since the instances only contain 8 choice alternatives. Furthermore, the hard datasets in the previous section were specifically constructed to be on the border of (non)rationalizability. Nonetheless, the large number of synthetic data sets that need to be evaluated for each instance make these computationally expensive. Using neither the starting set nor valid inequality pool, and using column generation with only the exact pricing solver, the instances take on average 15 minutes. Our algorithm without starting sets or valid inequality pool takes 1 minute on average

Table 5 shows that the valid inequality pool had a large impact on the computation time. Indeed, if we do not first check against the pool of notes inequalities, then every nonrationalizable data set requires solving at least one linear program and one exact pricing problem. Solving these is computationally expensive, whereas checking the valid inequality pool is computationally trivial in our analysis. Since the vast majority of sampled data sets were nonrationalizable, both the number of LPs and exact pricing problems that must be solved fell strongly. The starting sets provided a more modest speed-up. The re-use of generated variables did lower the number of pricing and restricted master problems that we must solve, both with and without valid inequality pools. It also lowered the number of pricing problems that required an exact approach. However, solving individual linear programs took longer, as they involved more variables. Total computation times for linear and pricing problems fell slightly. Note that the difference in total runtime is largely due to a difference in time spent on miscellaneous tasks, in this case resetting the master problem after evaluating each data set.

		Valid Inequality Pool							
		No				Yes			
		Time	Time Pricing	Time LP	CPLEX Calls	Time	Time Pricing	Time LP	CPLEX Calls
Re-use of	No	57.34	26.97	15.27	10533.16	1.29	0.19	0.33	38.81
variables	Yes	40.89	22.62	15.03	9997.47	1.06	0.18	0.22	37.13

Table 5: Comparison of computational results for valid inequality pools and re-use of variables.

We also note the following.

- Overall, instances for which many of the synthetic datasets are rationalizable required more computation time. This is in line with the result in the previous section which showed that the 'inside easy' instances took longer than the 'outside easy'. In the scenario without speed-ups, the instances for which more than 100 synthetic datasets were rationalizable took 66 seconds on average, while the instances with less than 10 rationalizable synthetic datasets took only 54 seconds. The valid inequalities strongly increased the difference (in relative computation time), in the scenario with all speed-ups the average times were 1.52 seconds and 0.92 seconds respectively.

- The stronger stopping condition had relatively little effect in these tests; in the scenario with all speed-ups it was used in only only one instance. For this instance, total computation time and the total number of calls to the exact pricing algorithm were higher when using the strong stopping condition. In this instance, the strong stopping condition was used on the third synthetic data set evaluated. The strong stopping condition terminated the column generation early. The valid inequality obtained here appears to be weaker than the one the algorithm found when the column generation was allowed to finish. It was only used to prove 428 synthetic datasets nonrationalizable, compared to 3503 datasets for the valid inequality obtained without the stronger stopping condition.

6 Conclusion

In this paper, we have presented an algorithm for testing models of probabilistic preferences (mixture models), based on column generation. This algorithm is capable of handling data sets of such size that the number of linear orders over all alternatives, and thus the number of variables in Formulation (1) would make the system of equalities prohibitive to solve. We have investigated the impact of different choices when it comes to implementing the algorithm: heuristic versus exact, generating a single solution versus generating many, using a strong stopping condition yes or no. The largest positive impact on the computation times comes from using the strong stopping condition. Furthermore, we have shown that the column generation algorithm is well-suited for testing large numbers of similar data sets, as variables can be re-used and the pricing objective function provides valid inequalities.

References

[1] M. Bazaraa, J. Jarvis, and H. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[2] H.D. Block and J. Marschak. Random orderings and stochastic theories of responses. *Contributions to probability and statistics*, 2:97–132, 1960.

[3] D. Cavagnaro and C. Davis-Stober. Transitive in our preferences, but transitive in different ways: An analysis of choice variability. *Decision*, 1(2):102, 2014.

[4] I. Charon and O. Hudry. A survey on the linear ordering problem for weighted or unweighted tournaments. *4OR*, 5(1):5–60, 2007.

[5] V. Chvátal. *Linear programming*. WH Freeman and Company, New York, 1983.

[6] C. Davis-Stober. A lexicographic semiorder polytope and probabilistic representations of choice. *Journal of Mathematical Psychology*, 56(2):86–94, 2012.

[7] J. Doignon, S. Fiorini, and G. Joret. Facets of the linear ordering polytope: a unification for the fence family through weighted graphs. *Journal of Mathematical Psychology*, 50(3):251–262, 2006.

[8] T. Dridi. Sur les distributions binaires associées à des distributions ordinales. *Mathématiques et Sciences Humaines*, 69:15–31, 1980.

[9] Julius Farkas. Theorie der einfachen ungleichungen. *Journal für Die Reine und Angewandte Mathematik*, 124:1–27, 1902.

[10] S. Fiorini. 0, 1/2-cuts and the linear ordering problem: Surfaces that define facets. *SIAM Journal on Discrete Mathematics*, 20(4):893–912, 2006.

[11] P. Gilmore and R. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[12] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.

[13] S. Hoderlein and J. Stoye. Revealed preferences in a heterogeneous population. *Review of Economics and Statistics*, 96(2):197–213, 2014.

[14] Y. Kitamura and J. Stoye. Nonparametric analysis of random utility models: Testing. Working paper, Cornell University, 2014.

[15] I. Klugkist and H. Hoijtink. The bayes factor for inequality and about equality constrained models. *Computational Statistics & Data Analysis*, 51(12):6367–6379, 2007.

[16] M. Laguna, R. Martí, and V. Campos. Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Computers & Operations Research*, 26(12):1217–1230, 1999.

[17] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[18] R. D. Luce. *Individual Choice Behavior: a Theoretical Analysis*. New York, NY: John Wiley, 1959.

[19] R. Martí and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*, volume 175 of *Applied Mathematical Sciences*. Springer-Verlag Berlin Heidelberg, 2011.

[20] R. Matzkin. Heterogeneous choice. *Econometric Society Monographs*, 43:75, 2007.

[21] D.L. McFadden and M.K. Richter. Stochastic rationality and revealed stochastic preference. In *Preferences, uncertainty, and optimality, essays in honor of Leo Hurwicz*, pages 161–186. Westview Press, 1990.

[22] N. Megiddo. Mixtures of order matrices and generalized order matrices. *Discrete Mathematics*, 19(2):177–181, 1977.

[23] M. Regenwetter, D. Cavagnaro, A. Popova, Y. Guo, C. Zwilling, S.H. Lim, and J.R. Stevens. Heterogeneity and parsimony in intertemporal choice. *Decision*, 2017.

[24] M. Regenwetter, J. Dana, and C. Davis-Stober. Testing transitivity of preferences on two-alternative forced choice data. *Frontiers in Psychology*, 1(148):1–15, 2010.

[25] M. Regenwetter, J. Dana, and C. Davis-Stober. Transitivity of preferences. *Psychological Review*, 118(1):42, 2011.

[26] M. Regenwetter and C. Davis-Stober. Behavioral variability of choices versus structural inconsistency of preferences. *Psychological Review*, 119(2):408, 2012.

[27] B. Smeulders. Recognizing single-peaked preferences on aggregated choice data. Technical report, KUL, 2014.

[28] A. Tversky. Intransitivity of preferences. *Psychological Review*, 76(1):31, 1969.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be